MODÜL 16

NESNE TABANLI PROGRAMLAMA

Şekil 16.1: *Bölümle ilgili örnek uygulamalara karekoddan ulaşabilirsiniz.*

16.1. Nesne Tabanlı Programlama (NTP) Nedir?

Nesne Tabanlı Programlama bir programlama paradigmasıdır. Bu yaklaşımda programlama mantığı gerçek hayattaki nesneler gibi tasarlanmıştır. Gerçek hayatta bir nesneye ihtiyacınız olduğunda onu alır ve kullanırsınız. Her nesnenin kendine ait özellikleri ve işlevleri vardır. Odanızdaki masa lambanız bozuldu ve yeni bir masa lambası almanız gerektiğini varsayınız. İçinde birçok aydınlatma cihazının olduğu büyük bir markete gittiniz. Raflarda çeşit değişik özelliklerde çeşit çeşit masa lambaları var. Tüm ürünlerin kullanılan malzeme, ışığın rengi, aydınlatma gücü, elektrik tüketimi, garanti süresi ve fiyatı gibi farklı özellikleri bulunmaktadır. Aynı zamanda bu cihazların açma, kapama, ışık ayarı gibi bazı işlevleri de bulunmaktadır. Nesneler (canlı veya cansız) tanımlanırken onların özelliklerinden ve işlevlerinden yararlanılır. Lambaların özellikleri farklı olsa da işlevleri değişse de hepsi ortak özellikleri olan bir

masa lambası formu (çizelge) ile tanımlanabilir. Bu formda her masa lambasına ait özellikler (olmayan bölümler için boş bırakılmış) ve bu cihazların işlevleri yer almaktadır. Hazırlanan genel form her masa lambasını temsil edebilecek özelliklerin ve işlevlerin bir listesini barındırmaktadır.

16.2. Sınıflar ve Nesneler

Masa lambası örneğinde olduğu gibi bir nesne ile ilgili değişik türde veriyi (formdaki veri: sayı, metin, liste) ihtiyaç duyulduğunda bunları yönetmek için nasıl bir yapı gereklidir? Bunun için bir liste, demet veya sözlük kullanıldığında bazı sıkıntılar yaşanabilir. Masa lambasının herhangi bir özelliğine bir değer atamak istediğinizi ya da bu özelliğe ilişkin veriyi almak istediğinizi düşününüz. Lamba ile ilgili değişik veri türlerini yönetebileceğiniz bir yapıya ihtiyaç vardır. Lambanın özelliklerini ve işlevlerini yerine getirecek değişkenleri ve fonksiyonları tanımlamak gerekir. Eğer bu ihtiyaçlar NTP yaklaşımı ile ele alınmazsa tüm bu özellikler ve işlevler için farklı ve dağınık yapılar oluşacaktır. NTP, bir nesne olarak yapının tüm özelliklerinin ve işlevlerinin (metot) bir yerde toplanmasına ve kullanılmasına olanak verir. Bir sınıf tanımlandığı zaman aslında bir model oluşturulur. Bu model o sınıfa ait tüm nesneleri temsil eden ve aslında hiçbiri olmayan genel bir model, bir şablondur. Bu modeli kullanarak o sınıftaki tüm nesneler türetilebilir. Bir sınıftan bir nesne türetildiğinde yapıya ilişkin modelin bir kopyası oluşur. Nesne ile sınıfın özellikleri ve işlevleri kullanılıp özelleştirilebilir.

NTP'de oluşturulan yapılar nesneler tarafından temsil edilir. Bu nesneler kullanıcı tarafından veya başkaları tarafından oluşturulan diğer nesnelerle etkileşime girebilirler. NTP gerçek dünyada nesneler arasındaki etkileşim gibi programlama yapılarındaki etkileşimleri de modellemeye çalışır. Program akışına göre nesneler diğer nesnelerle etkileşime girerek özelliklerini ve işlevlerini o nesnelerin erişimine açabilir. Başka bir nesne, eriştiği nesneden aldığı değeri kullanarak yeni işlemler yapabilir. Bu sonuç bir argüman (girdi) olarak kullanılıp nesnenin kendi işlevlerinden biri çalıştırılabilir. Böylelikle gereksinim duyulan bir program için programcının kendi oluşturduğu veya önceden oluşturulmuş nesneler uygun şekilde bir arada kullanılabilir. Python'da tüm yapılar sınıflar içinde oluşturulur.

Python'da kullanılan veri türlerinin her biri bir sınıftır.

Örnek 1

type( ) komutunu kullanarak veri türlerini görüntüleyerek bunu görebilirsiniz.

```
print(type(int))
print(type(str))
print(type(list))
print(type(tuple))
print(type(dict))
<class 'type'>
<class 'type'>
<class 'type'>
<class 'type'>
<class 'type'>
```

16.2.1. Bir Sınıf Tanımlama

Örnek 2

Basit bir sınıf tanımlayınız. Bir sınıf class SinifAdi: veya class SinifAdi(): şeklinde tanımlanır.

Blok yapısı içinde yer alır.

Özellikler ve metotlar sınıfın içinde tanımlanır.

```
class OrnekSinif:
  __doc__='Bu sınıf örnek amaçlı oluşturulmuştur.'
  def __init__(self):
    print ('Merhaba Sınıf')
```

Sınıfı tanımlarken class OrnekSinif(): olarak yapabilirsiniz. Sınıftan bir nesne oluşturabilirsiniz.

16.2.2. Sınıftan Bir Nesne Oluşturma

Örnek 3

Sınıflardan nesne türetme (instance) örnek oluşturma olarak adlandırılır.

```
nesnem=OrnekSinif()
print (nesnem)
Merhaba Sınıf
<__main__.OrnekSinif object at 0x7fcbe181acc0>
```

Nesnenizi oluşturdunuz. print( ) kullandığınızda nesnenin ait olduğu sınıfın adı ve hafızadaki yeri yazdırılacaktır. Sınıfın açıklamasına bakmak için "nesnem.__doc__"u kullanabilirsiniz.

Örnek 4

```
nesnem.__doc__
'Bu sınıf örnek amaçlı oluşturulmuştur.'
```

16.2.3. Yapıcı "__init__()" Fonksiyonu

Bir sınıftan bir nesne oluşturulduğunda otomatik olarak çalışan bir fonksiyondur. NTP'de yapıcı (constructor) fonksiyon olarak adlandırılan bu fonksiyonlar nesne için zorunlu parametreleri işler ve nesnenin oluşturulmasını sağlar.

Örnek 5

Yapıcı fonksiyonu self olmadan tanımlayınız.

```
class OrnekSinif:
  def __init__():
    print ('Merhaba Sınıf')
nesnem=OrnekSinif()
TypeError                                 Traceback (most recent call last)
<ipython-input-51-dca29725c758> in <module>()
      2   def __init__():
      3     print ('Merhaba Sınıf')
----> 4 nesnem=OrnekSinif()

TypeError: __init__() takes 0 positional arguments but 1 was given
```

Kodu "self" olmadan kullandığınızda yukarıdaki hata ile karşılaşırsınız.

Örnek 6

Verilen metin ifadelerini istenilen kadar yazan bir sınıf tanımlayınız.

Sınıf için bazı zorunlu parametreler oluşturulacaktır.

```
class MetinSinifi:
  def __init__(self, ifade, tekrarSayisi, kacisKarakteri):
    print ((ifade+kacisKarakteri)*tekrarSayisi)
nesnem=MetinSinifi('Sınıf oluşturduk.',5, '\n')
Sınıf oluşturduk.
Sınıf oluşturduk.
Sınıf oluşturduk.
Sınıf oluşturduk.
Sınıf oluşturduk.
```

16.2.4. Aşırı Yükleme (Over Loading)

Bilindiği gibi fonksiyonlarda, kullanmak zorunda olunan parametreler belirtilmektedir. NTP'de bir fonksiyon aynı isimle farklı parametrelerle tanımlanabilir. Python'da bir sınıf içinde aynı isimde birden fazla fonksiyon tanımlamak yani aşırı yükleme yapmak (method overloading) hataya neden olur.

Bu durumun üstesinden gelmek için fonksiyonlar konusunda belirtildiği gibi bazı parametreler için varsayılan değerler atanabilir. Aşağıdaki kodda kacisKarakteri varsayılan olarak '\n' verilmiştir. Böylece sınıf kullanıldığında bu parametreye bir argüman verilmezse hata ile karşılaşılmaz.

Örnek 7

Sınıfı farklı sayıda argümanlarla kullanabilirsiniz.

```
class MetinSinifi:
  def __init__(self, ifade, tekrarSayisi, kacisKarakteri='\n'):
    print ((ifade+kacisKarakteri)*tekrarSayisi)
nesnem=MetinSinifi('Sınıf oluşturduk.',5)
Sınıf oluşturduk.
Sınıf oluşturduk.
Sınıf oluşturduk.
Sınıf oluşturduk.
Sınıf oluşturduk.
```

Bir sınıf içindeki bütün fonksiyonlar "self" parametresini zorunlu olarak alırlar. Bu sayede fonksiyonlar sınıf içindeki tüm fonksiyonlara ve değişkenlere erişim sağlar. "self" yerine başka bir parametre adını kendiniz verebilirsiniz ancak bu kullanım standarttır. Kullanırken self parametresine argüman verilmez. Sınıfınızı "self" olmadan yazdığınızda hata ile karşılaşırsınız.

Örnek 8

Sınıf oluştururken verdiğiniz argümanlara erişmek için aşağıdaki kodları kullanabilirsiniz.

```
class MetinSinifi:
  def __init__(self, ifade, tekrarSayisi, kacisKarakteri='\n'):
    self.ifade=ifade
    self.tekrarSayisi=tekrarSayisi
    print ((ifade+kacisKarakteri)*tekrarSayisi)
```

Sınıfınızdan bir nesne oluşturarak parametrelerine tekrar erişebilirsiniz. Zorunlu parametrelere değer vererek nesne tanımlayabilirsiniz. Sonrasında fonksiyon içindeki değişkenlere erişim sağlayabilirsiniz.

MetinSinifi'dan tanımladığınız metinNesnesi.ifade ve metinNesnesi.tekrarSayisi argüman olarak verdiğiniz değerlere erişebilirsiniz.

NOT

Argümanlarınızı tanımlarken varsayılan değerli argümanlar, zorunlu argümanlardan sonra gelmelidir.

Örnek 9

```
metinNesnesi=MetinSinifi('Müdür', 3)
print ('Yazılan Metin: ', metinNesnesi.ifade)
print('Tekrar Sayısı: ', metinNesnesi.tekrarSayisi)
Müdür
Müdür
Müdür
Yazılan Metin:  Müdür
Tekrar Sayısı:  3
```

Program yazarken size bir masa lambası lazım olduğunda o sınıftan bir nesne oluşturur (örneğin lambam) bunun tüm özelliklerini belirtebilir, değiştirebilir ve bu sınıfa tanımlı işlevleri kullanabilirsiniz. Böylece hayalinizdeki çok ısınmayan, az enerji tüketen, ışık şiddetini ayarlayabildiğiniz ve istediğiniz malzemeden

yapılmış bir masa lambası oluşturabilirsiniz. lamba1.isikRengi='Sari' lamba1.isikSiddeti='100' lamba1.aydinlat()

aydinlat() lamba1 nesnesine ait bir metot yani fonksiyondur. Özellikler (isikRengi, isikSiddeti) ise aslında değerleri tutan değişkenlerdir.

Örnek 10

Bir Masa Lambası sınıfı oluşturunuz.

```
class MasaLambasi:
#Örnek bir sınıf
  isikRengi='sarı' # özellik
  isikDurum=False # özellik
  def isikAc(self): # metot, fonksiyon
    self.isikDurum=True
    return 'Aydınlık'
```

Örnek 11

Sınıfınızı adıyla çağırabilirsiniz.

```
MasaLambasi.isikRengi
'sarı'
```

Örnek 12

Sınıfınızdan bir nesne türetebilir ve sınıfınızda tanımladığınız özelliklere erişebilirsiniz.

```
lambam=MasaLambasi()
print (lambam.isikAc())
print (lambam.isikDurum)
Aydınlık
True
```

Örnek 13

Oluşturduğunuz nesneyi silmek için del( ) fonksiyonunu kullanabilirsiniz.

```
del(lambam)
print(lambam.isikDurum)
#Nesne silindiği için hata ile karşılaşılır.
NameError                 Traceback (most recent call last)
<ipython-input-67-c18bef2cc0fd> in <module>()
----> 1 del(lambam)
      2 print(lambam.isikDurum)
      3 #Nesne silindiği için hata ile karşılaşılır.

NameError: name 'lambam' is not defined
```

Örnek 14

Masa lambası sınıfınızı yeniden tanımlayınız. Bu sınıfa farklı özellikler ve işlevler ekleyiniz.

```
class MasaLambasi:
  __doc__='Masa Lambası Sınıfı'
  isikDurum=False
  def __init__(self,    isikSiddeti, isikRengi='sarı', garantiSuresi=2):
    self.isikSiddeti=isikSiddeti
  def isikAc(self):
    self.isikDurum=True
  def isikKapat(self):
    self.isikDurum=False
  def isigiArtir(self, artirmaMiktari):
    self.isikSiddeti+=artirmaMiktari
```

Örnek 15

Sınıfınızdan bir nesne türetiniz. Bunun için parametrelere argümanları yani değerleri girebilirsiniz.

```
lambam=MasaLambasi(60, garantiSuresi=3)
```

Argümanları parametre sırasına göre verebileceğiniz gibi argüman adlarını kullanarak da değer verebilirsiniz.

Örnek 16

Nesnenin özelliklerini ve işlevlerini kullanabilirsiniz.

```
lambam.isikAc() #şimdi ışığı açalım
print ('Işık Açık mı? ', lambam.isikDurum) #Işığın durumu
Işık Açık mı?  True
```

Örnek 17

Işığın şiddetini arttırarak son durumu kontrol edebilirsiniz.

```
lambam.isigiArtir(5)
print('Işık Şiddeti:', lambam.isikSiddeti)
Işık Şiddeti: 65
```

Örnek 18

Işığı kapatabilirsiniz.

```
lambam.isikKapat()
print('Işık Açık mı?', lambam.isikDurum)
Işık Açık mı? False
```

Masa Lambası nesnesini kullanarak programınızda ihtiyaç duyduğunuz masa lambasının tüm özelliklerini ve tüm işlevlerini tek bir yapı içinde (nesne olarak) kontrol edebilirsiniz. Bu sınıfı aynı şekilde bir bütün olarak diğer projelerinizde de kullanabilirsiniz.

Örnek 19

Aşağıdaki örnekte öğrenci verilerini ve öğrencilere ait işlemleri yaptığınız bir sınıf oluşturunuz.

```
class Ogrenci():
  __doc__='Öğrenci sınıfı'
  ogrenciler=[]
  def __init__(self, ogrenciAdiSoyadi):
    self.ogrenciAdiSoyadi = ogrenciAdiSoyadi
    self.dersleri = []
    self.sinavlar=[]
    self.ogrenciEkle(self.ogrenciAdiSoyadi)

  def ogrenciEkle(self, adSoyad):
    self.ogrenciler.append(adSoyad)
    print('{} adlı kişi öğrencilere eklendi.'.format(adSoyad))

  def ogrenciListesiYazdir(self):
    print('Öğrenci listesi')
    for ogrenci in self.ogrenciler:
      print(ogrenci)

  def dersEkle(self, dersAdi):
    self.dersleri.append(dersAdi)

  def ogrencininDersleri(self):
    print('{} adlı kişinin dersleri:'.format(self.ogrenciAdiSoyadi))
    for ders in self.dersleri:
      print(ders)

  def sinavPuaniEkle(self, sinavPuani):
    self.sinavlar.append(sinavPuani)

  def ogrencininSinavPuanlari(self):
    print('{} adlı kişinin sınav sonuçları:'.format(self.ogrenciAdiSoyadi))
    for sinav in self.sinavlar:
      print(sinav)
```

Örnek 20

Bir sınıfın içeriğine bakmak için "dir(sinifAdi)" kullanabilirsiniz. Standart parametrelerin yanı sıra tanımlı olan fonksiyonların listesine erişebilirsiniz.

```
dir(Ogrenci)
['__class__', '__delattr__', '__dict__', '__dir__','__doc__','__eq__','__
format__','__ge__', '__getattribute__', '__gt__', '__hash__',
'__init__','__init_subclass__','__le__','__lt__','__module__','__ne__','__
new__','__reduce__','__reduce_ex__', '__repr__', '__setattr__', '__
sizeof__', '__str__', '__subclasshook__', '__weakref__', 'dersEkle',
'ogrenciEkle', 'ogrenciListesiYazdir','ogrenciler','ogrencininDersleri',
'ogrencininSinavPuanlari','sinavPuaniEkle']
```

Örnek 21

Sınıfınızdan bir nesne türeterek bir öğrenci ekleyebilirsiniz.

```
ogrenciNesnesi=Ogrenci('Murat Çalışkan')
Murat Çalışkan adlı kişi öğrencilere eklendi
```

Örnek 22

Öğrenci sayısına bakabilirsiniz.

```
print ('Öğrenci Sayısı:', len(ogrenciNesnesi.ogrenciler))
```

Örnek 23

Öğrencilere dersleri ve sınav puanlarını ekleyebilirsiniz.

```
ogrenciNesnesi.dersEkle('Türkçe')
ogrenciNesnesi.sinavPuaniEkle(95)
ogrenciNesnesi.dersEkle('Matematik')
ogrenciNesnesi.sinavPuaniEkle(100)
ogrenciNesnesi.dersEkle('Fen Bilgisi')
ogrenciNesnesi.sinavPuaniEkle(100)
```

Örnek 24

Öğrencinin derslerini ve sınav puanlarını listeleyebilirsiniz.

```
ogrenciNesnesi.ogrencininDersleri()
ogrenciNesnesi.ogrencininSinavPuanlari()
Murat Çalışkan adlı kişinin dersleri:
Türkçe
Matematik
Fen Bilgisi
Murat Çalışkan adlı kişinin sınav sonuçları:
95
100
100
```

Örnek 25

Şimdi bir öğrenci daha ekleyiniz ve öğrenci listesini yazdırınız.

```
ogrenciNesnesi.ogrenciEkle('Ahmet Birinci')
Ahmet Birinci adlı kişi öğrencilere eklendi.
len(ogrenciNesnesi.ogrenciler)
2
```

Örnek 26

Öğrenci listesini görebilirsiniz.

```
ogrenciNesnesi.ogrenciListesiYazdir()
Öğrenci listesi
Murat Çalışkan
Ahmet Birinci
```

16.3. Sınıf Metotları

Sınıf tanımlarken sınıftan bir nesne türetmeden fonksiyonlarına erişmek için "@classmethod" kullanılır.

Örnekte "Ogrenciler" sınıfı oluşturdunuz ve içinde öğrenci verilerinin bulunduğu bir liste var. Fonksiyon, bir sınıf metodu olarak tanımlanırken satır başına "@classmethod" eklenir ve "self" tanımında olduğu gibi fonksiyona da "cls" eklenir. Bu parametre adı değiştirilebilir ancak standart olarak bu şekilde kullanılmaktadır. Sınıf metotları sınıftan nesne türetmeden doğrudan sınıfın metotlarına erişme olanağı sağlar. Aşağıdaki sınıfta __init__ sınıf içindeki metotlar kullanılabilmektedir. Aynı zamanda bu metotlara sınıf üzerinden doğrudan erişilebilmektedir.

Örnek 27

```
class Ogrenciler():
    def __init__(self, sorgu=None, sirasi=None):
      self.ogrenciListe = [('Murat Çalışkan', 11, 131), ('Ahmet Birinci', 11, 124),
('Emre Hızlı', 12, 135), ('Murat Çalışkan', 12, 155)]
      if not sorgu and not sirasi:
        listem = self.ogrenciListe
      else:
        listem = [li for li in self.ogrenciListe if sorgu == li[sirasi]]
      for i in listem:
        print(*i, sep=', ')

    @classmethod #Bir sınıf metodu tanımlanıyor
    def ogrenciAdindanSorgula(cls, adi): # self gibi burada da cls parametresi
kullanılıyor
        cls(adi, 0)

    @classmethod
    def ogrenciSinifindanSorgula(cls, sinifi):
        cls(sinifi, 1)

    @classmethod
    def ogrenciNumarasindanSorgula(cls, numarasi):
        cls(numarasi, 2)
```

Örnek 28

Doğrudan sınıfın metotlarını çağırarak ve adını verdiğiniz bir öğrencinin bilgilerini listeleyiniz.

```
Ogrenciler.ogrenciAdindanSorgula('Murat Çalışkan')
Murat Çalışkan, 11, 131
Murat Çalışkan, 12, 155
```

Örnek 29

Belirli bir sınıftaki öğrencilerin listesini alınız.

```
Ogrenciler.ogrenciSinifindanSorgula(12)
Emre Hızlı, 12, 135
Murat Çalışkan, 12, 155
```

Örnek 30

Öğrenci numarasından sorgulama yapınız.

```
Ogrenciler.ogrenciNumarasindanSorgula(131)
Murat Çalışkan, 11, 131
```

Örnek 31

Tüm öğrencileri listelemek için Ogrenciler() argüman kullanmadan çağırmalısınız. Bunun için sınıfınızdan bir nesne türetiniz.

```
tumOgrenciListe=Ogrenciler()
Murat Çalışkan, 11, 131
Ahmet Birinci, 11, 124
Emre Hızlı, 12, 135
Murat Çalışkan, 12, 155
```

16.4. Kapsülleme

Sınıf içindeki metotlara ve özelliklere erişimleri kısıtlamak için kapsülleme yöntemi kullanılır. Bir sınıf tanımlanırken kapsülleme ile erişilmesi, değiştirilmesi istenmeyen veya sadece kısıtlı erişim verilmek istenilen özellikler ve/veya metotlar tanımlanabilir. Bunun için istenilen metodun veya özelliğin başına "__" eklenmelidir.

Örnek 32

Örnekte self.__TCNo=TCNo kullanımı öğrencinin kimlik numarasına dışarıdan erişimi engeller.

```
class Ogrenciler:
  def __init__(self,adi,yasi,TCNo):
    self.adi=adi
    self.yasi=yasi
    self.__TCNo=TCNo
  def ogrenciYaz(self):
    print('Öğrenci Bilgileri')
    print(self.adi)
    print(self.yasi)
    print(self.__TCNo)
ogrenci=Ogrenciler('Murat Çalışkan',17,54639876211)
ogrenci.ogrenciYaz()
Öğrenci Bilgileri
Murat Çalışkan
17
54639876211
```

Örnek 33

Öğrenci bilgilerine doğrudan erişmeyi deneyiniz.

```
print (ogrenci.adi)
print (ogrenci.yasi)
Murat Çalışkan
17
```

Örnek 34

Öğrencinin adı ve yaşı değişkenleri gizli olmadığı için sınıf dışından doğrudan ulaşılabilir. Öğrencinin kimlik numarasına erişmeye çalışınız.

```
print(ogrenci.__TCNo)
AttributeError                          Traceback (most recent call last)
<ipython-input-212-62da05ea5506> in <module>()
----> 1 print(ogrenci.__TCNo)

AttributeError: 'Ogrenciler' object has no attribute '__TCNo'
```

Örnek 35

Sınıf içinde özel olarak tanımlanmış bu tür özelliklere erişmek için "set" ve "get" metodları (diğer NTP dillerinde de farklı adlarla) bulunur. "get" özellikleri okumak "set" ise değiştirmek için kullanılır.

```
class Ogrenciler:
  def __init__(self,adi,yasi,TCNo):
    self.adi=adi
    self.yasi=yasi
    self.__TCNo=TCNo

  def ogrenciYaz(self):
    print('Öğrenci Bilgileri')
    print(self.adi)
    print(self.yasi)
    print(self.__TCNo)

  def get_TCNo(self):
    return str(self.__TCNo)[7:]
```

Yukarıda tanımlanan def get_TCNo(self): ile kimlik numarasının son 4 hanesi döndürülmüştür. Kapsülleme sınıfın içindeki özellikleri ve metotları korumayı sağlamaktadır. Kapsülleme aynı zamanda verinin güvenliğini sağlar.

Örnek 36

```
ogrenci=Ogrenciler('Sezai', 43, 12345678901)
print(ogrenci.get_TCNo())
8901
```

Örnek 37

"get" metodu ile yazma özelliği verebilir, "set" metodu ile de değişkene yeni değer atayabilirsiniz.

```
class Ogrenciler:
  def __init__(self,adi,yasi,TCNo):
    self.adi=adi
    self.yasi=yasi
    self.__TCNo=TCNo
   def ogrenciYaz(self):
    print('Öğrenci Bilgileri')
    print(self.adi)
    print(self.yasi)
    print(self.__TCNo)
    def get_TCNo(self):
    return str(self.__TCNo)[7:]

  def set_TCNo(self,TcNoDuzelt):
    self.__TCNo=TcNoDuzelt
```

Örnek 38

Yanlış girilen kimlik numarasının düzeltildiğin varsayınız. Öncelikle bir öğrenci tanımlayınız.

```
ogrenci=Ogrenciler('Kaan Emre', 33, 23142343655)
print('Öğrenci TC No son 4 hane: ', ogrenci.get_TCNo())
Öğrenci TC No son 4 hane:  3655
```

Örnek 39

Öğrencinin kimlik numarasını düzeltebilir ve tekrar yazdırabilirsiniz.

```
ogrenci.set_TCNo(12309768687)
print ('Öğrenci TC No son 4 hane: ', ogrenci.get_TCNo())
Öğrenci TC No son 4 hane:  8687
```

16.5. Kalıtım (Miras Alma)

Kalıtım bir sınıfın başka bir sınıfın metotlarını ve özelliklerini kendi bünyesine almasıdır. Bu metot ve özelliklerin devralındığı (Miras alma da denir.) sınıfa üst sınıf, devralan sınıfa ise alt sınıf denir. Alt sınıflar, üst sınıfın tüm metotlarını ve özelliklerini devralır, ancak farklı özellikler ve metotlar da ekleyebilir. En temel sınıf türü, genellikle diğer tüm sınıfların ebeveynleri olarak miras aldığı bir nesnedir. Örneğin bir "kedi" sınıfının metotları ve özellikleri neler olabilir? Her kedinin bir adı, yaşı ve cinsi vardır. Buna ek olarak her kedi cinsinin farklı özellikleri de vardır. Her kedi cinsi, üst sınıf olan kedi sınıfının alt sınıfıdır. Üst sınıflara "ebeveyn (parent) sınıf" alt sınıflara ise "çocuk (child) sınıf" denir.

Örnek 40

Bu örnekte bir kedi üst sınıfı tanımlayınız.

```
#Üst Sınıf
class Kedi:
    tur = 'Memeli'

    def __init__(self, adi, yasi):
      self.adi = adi
      self.yasi = yasi

    def ozellikler(self):
      return "{} {} yaşında".format(self.adi, self.yasi)

    def miyavla(self, sesi):
      return "{}  {} diye miyavlar".format(self.adi, sesi)
```

Örnek 41

Kedi üst sınıfından farklı türdeki kediler için alt sınıflar tanımlayınız.

```
# Kedi alt sınıfından miras alan alt sınıf
class PersKedisi(Kedi):
  def tuyleri(self, tuyuNasil):
    return "{} tüyleri {}".format(self.adi, tuyuNasil)

class VanKedisi(Kedi):
  def gozRengi(self, gozRengiNe):
    return "{} gözleri {}".format(self.adi, gozRengiNe)
```

Alt sınıftan "Pisi" adında ve "4" yaşında bir kedi türetiniz.

```
pisi = VanKedisi('Pisi', 4)
print(pisi.ozellikler())
print(pisi.gozRengi('Mavi ve kahverengi'))
Pisi 4 yaşında
Pisi gözleri Mavi ve kahverengi
```

Örnek 42

"PersKedisi" ve "VanKedisi" alt sınıflarını "Kedi" üst sınıfından metotları ve özellikleri devralarak oluşturdunuz. Bu iki alt sınıfa kendilerine özgü yeni özellikler ve metotlar tanımlayarak örneğe devam ediniz.

```
persia= PersKedisi('Persia', 4)
print(persia.ozellikler())
print(persia.miyavla('meaaaw'))
print(persia.tuyleri('Yumuşak'))
Persia 4 yaşında
Persia meaaaw diye miyavlar
Persia tüyleri Yumuşak
```

Örnek 43

Diğer alt sınıfın da kendine ait özelliklerini ve üst sınıftan devraldığı özellik ve metotları çağırabilirsiniz. Üst sınıftan devralınan metotları ve özellikleri geçersiz (overridding) kılabilirsiniz.

```
class VanKedisi(Kedi):
  def gozRengi(self, gozRengiNe):
    return "{} gözleri {}".format(self.adi, gozRengiNe)
  def ozellikler(self):
    return "Türü: Van kedisi adı:{} ve {} yaşında".format(self.adi, self.yasi)
    #overridding
pisi = VanKedisi('Pisi', 4)
print(pisi.ozellikler())
print(pisi.gozRengi('Mavi ve kahverengi'))
Türü: Van kedisi adı:Pisi ve 4 yaşında
Pisi gözleri Mavi ve kahverengi
```

Yukarıdaki örnekte özellikler metodunu geçersiz kılınarak üst sınıftan farklı bir değer döndürülmüştür.

16.6. Bölüm Sonu Örnekleri

1. Bir araba sınıfı tanımlayınız.

Özellikler:

marka, model, kapı sayısı, rengi, fiyatı, motor seri numarası (gizli özellik). Marka ve model dışındaki tüm özelliklerin varsayılan değeri yok (None), kapı sayısı varsayılan değeri=2 olarak tanımlanacaktır. Arabanın özelliklerini yazdıran bir metot tanımlayınız.

2. Bu araba sınıfından sarı renkli 2 kapılı 1993 model bir "Marka 00" marka araba türetiniz. Motor şase numarası 123456789 olsun. Daha sonra bu arabanın özelliklerini yazdırınız.

3. Motor şase numarasını okumak ve değiştirmek için sınıfı düzenleyiniz. Şase numarasını değiştirerek ekrana yazdırınız.

4. Araba sınıfından kalıtım yoluyla bir sınıf üretiniz ve gazVer metodunda ekrana 'Gaza basıldı' yazmasını sağlayınız. Bu alt sınıfa çelik jant özelliği ekleyiniz.

Cevaplar

1.
```
class Araba:
  __doc__='Araba Sınıfı'
  isikDurum=False
  def __init__(self,    marka, model, kapiSayisi=2, rengi=None, fiyat=None,
motorSeriNo=None):
    self.marka=marka
    self.model=model
    self.kapiSayisi=kapiSayisi
    self.rengi=rengi
    self.__fiyat=fiyat
    self.__motorSeriNo=motorSeriNo
  def arabaOzellikleri(self):
    print ('{} model  {} kapılı  {} renkli {} marka araç'.format(self.model,
self.kapiSayisi, self.rengi, self.marka))
```

2.
```
arabam=Araba('Marka 00', 1993,2,'sarı',motorSeriNo=123456789)
arabam.arabaOzellikleri()
1993 model  2 kapılı  sarı renkli Marka 00 marka araç
```

3.
```
class Araba:
  __doc__='Araba Sınıfı'
  isikDurum=False
  def __init__(self,    marka, model, kapiSayisi=2, rengi=None, fiyat=None,
motorSeriNo=None):
    self.marka=marka
    self.model=model
    self.kapiSayisi=kapiSayisi
    self.rengi=rengi
    self.__fiyat=fiyat
    self.__motorSeriNo=motorSeriNo
  def arabaOzellikleri(self):
    print ('{} model  {} kapılı  {} renkli {} marka araç motor seri no: {}'.
format(self.model,  self.kapiSayisi, self.rengi, self.marka, self.__motorSeriNo))
  def get_motorSeriNo(self):
    return self.__motorSeriNo
  def set_motorSeriNo(self, motorSeriNoDuzeltme):
    self.__motorSeriNo=motorSeriNoDuzeltme

arabam=Araba('Marka 123', 1981)
arabam.arabaOzellikleri()
arabam.set_motorSeriNo(123456) #yeni seri numara değerini verdik
print(arabam.get_motorSeriNo())
1981 model  2 kapılı  None renkli Marka 123 marka araç motor seri no: None
123456
```

4.
```
class ArabaSinifim(Araba):
  çelikJant=True
  def gazVer(self):
    print (self.marka, 'Durum: Gaza basıldı')

tomofil=ArabaSinifim('Marka 456', 1992)
tomofil.gazVer()
tomofil.arabaOzellikleri()
Marka 456 Durum: Gaza basıldı
1992 model  2 kapılı  None renkli Marka 456 marka araç motor seri no: None
```